

# A NONLINEAR GMRES OPTIMIZATION ALGORITHM FOR CANONICAL TENSOR DECOMPOSITION

H. DE STERCK\*§

**Abstract.** A new algorithm is presented for computing a canonical rank- $R$  tensor approximation that has minimal distance to a given tensor in the Frobenius norm, where the canonical rank- $R$  tensor consists of the sum of  $R$  rank-one components. Each iteration of the method consists of three steps. In the first step, a tentative new iterate is generated by a stand-alone one-step process, for which we use alternating least squares (ALS). In the second step, an accelerated iterate is generated by a nonlinear generalized minimal residual (GMRES) approach, recombining previous iterates in an optimal way, and essentially using the stand-alone one-step process as a preconditioner. In particular, the nonlinear extension of GMRES is used that was proposed by Washio and Oosterlee in [ETNA Vol. 15 (2003), pp. 165-185] for nonlinear partial differential equation problems. In the third step, a line search is performed for globalization. The resulting nonlinear GMRES (N-GMRES) optimization algorithm is applied to dense and sparse tensor decomposition test problems. The numerical tests show that ALS accelerated by N-GMRES may significantly outperform both stand-alone ALS and a standard nonlinear conjugate gradient optimization method, especially when highly accurate stationary points are desired for difficult problems. The proposed N-GMRES optimization algorithm is based on general concepts and may be applied to other nonlinear optimization problems.

**Key words.** canonical tensor decomposition, alternating least squares, GMRES, nonlinear optimization

**AMS subject classifications.** 15A69 Multilinear algebra, 65F10 Iterative methods, 65K10 Optimization, 65F08 Preconditioners for iterative methods

**1. Introduction.** In this paper, we present a new algorithm for computing a canonical rank- $R$  tensor approximation that has minimal distance to a given tensor in the Frobenius norm, where the canonical rank- $R$  tensor consists of the sum of  $R$  rank-one components. As one of its components, the optimization algorithm uses a nonlinear version of the generalized minimal residual (GMRES) method that was originally proposed for iteratively solving systems of linear equations [15]. More specifically, we use the nonlinear extension of GMRES that was developed by Washio and Oosterlee [18] for nonlinear partial differential equation (PDE) systems, and apply it to the tensor optimization problem, with the purpose of efficiently driving the gradient of the objective function to zero in a process that uses iterate recombination. We apply this nonlinear GMRES acceleration to the alternating least squares (ALS) method, and combine it with a line search for globalization. We perform numerical tests to investigate the performance of the resulting nonlinear GMRES (N-GMRES) optimization algorithm for canonical tensor approximation.

$N$ -way tensor  $\mathcal{T} \in \mathbb{R}^{I_1 \times \dots \times I_N}$  is an  $N$ -dimensional array of size  $I_1 \times \dots \times I_N$  [9]. The size of mode  $n$  is  $I_n$  ( $n = 1, \dots, N$ ). Let  $\mathcal{A}_R \in \mathbb{R}^{I_1 \times \dots \times I_N}$  be a canonical rank- $R$  tensor, given by

$$\mathcal{A}_R = \sum_{r=1}^R \mathbf{a}_r^{(1)} \circ \dots \circ \mathbf{a}_r^{(N)} = [\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)}]. \quad (1.1)$$

Canonical tensor  $\mathcal{A}_R$  is a sum of  $R$  rank-one tensors, with the  $r$ th rank-one tensor composed of the outer product of  $N$  column vectors  $\mathbf{a}_r^{(n)} \in \mathbb{R}^{I_n}$ ,  $n = 1, \dots, N$ . For

\*Department of Applied Mathematics, University of Waterloo, Waterloo, Ontario, Canada

§hdesterck@uwaterloo.ca

each mode  $n = 1, \dots, N$ , the  $R$  vectors  $\mathbf{a}_r^{(n)}$ ,  $r = 1, \dots, R$ , form the columns of the mode- $n$  factor matrix  $\mathbf{A}^{(n)}$ , and the double-bracket notation on the right of (1.1) is used to denote the canonical tensor by the factor matrices.

This paper concerns numerical algorithms for the following optimization problem:

OPTIMIZATION PROBLEM I:

given tensor  $\mathcal{T} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ , find rank- $R$   
canonical tensor  $\mathcal{A}_R \in \mathbb{R}^{I_1 \times \dots \times I_N}$  that minimizes

$$f(\mathcal{A}_R) = \frac{1}{2} \|\mathcal{T} - \mathcal{A}_R\|_F^2. \quad (1.2)$$

Here,  $\|\cdot\|_F$  denotes the Frobenius norm of the  $N$ -dimensional array, which is the square root of the sum of the squares of all the array elements.

We now briefly recall some properties of canonical tensor decomposition, mainly referring to review article [9], which also contains extensive references to original papers that the reader may consult for more detail. The exact decomposition of a data tensor  $\mathcal{T}$  into a canonical tensor is often called a CP decomposition of the tensor, with the C standing for ‘CANDECOMP’ and the ‘P’ standing for ‘PARAFAC’, after the names originally given to this decomposition in early papers on the subject [3, 7]. The smallest number of rank-one components that generate  $\mathcal{T}$  as their sum is called the tensor rank of tensor  $\mathcal{T}$  [9]. If, rather than an exact decomposition,  $\mathcal{T}$  is *approximately* decomposed into a low-rank canonical tensor  $\mathcal{A}_R$  of specified rank  $R$  that is smaller than the rank of  $\mathcal{T}$ , then the resulting tensor  $\mathcal{A}_R$  is called an approximate CP decomposition. CP decompositions are used for data analysis in many applications such as chemometrics, signal processing, neuroscience and web analysis. Many criteria exist for specifying the type of approximation that is sought for approximate CP decompositions (see, e.g., [17]). In this paper, we focus on the specific (and practically relevant) case of computing an approximate CP decomposition  $\mathcal{A}_R$  that minimizes the Frobenius distance between the data tensor and  $\mathcal{A}_R$ , i.e., we seek to minimize objective function (1.2). Contrary to the case of best rank- $R$  matrix approximation, the rank-one terms of the best rank- $R$  CP tensor approximation cannot be solved for sequentially but must be found simultaneously, since a best rank- $R$  CP approximation cannot be obtained by truncating a best rank- $S$  approximation with  $S > R$  [9].

It is well-known that Optimization Problem I is a non-convex optimization problem, and as such may exhibit multiple local minima. In the form given above, its local minima are not isolated, since there is a scaling indeterminacy in each of the rank-one terms: there is ample freedom to rescale the vectors  $\mathbf{a}_r^{(n)}$  in (1.1) without changing the rank-one product. In our approach, we deal with this by normalizing the  $\mathbf{a}_r^{(n)}$  in a specific way, to be explained below. There is also a permutation indeterminacy in the order of the rank-one terms, which we deal with by imposing a specific order, also to be explained below. Even when the scaling and permutation indeterminacies are removed, CP optimization may still exhibit multiple local minima for some problems, and depending on the initial guess, iterative methods for approximate CP decomposition may converge to different stationary points. It is also possible that the best rank- $R$  approximation does not exist, which may happen when some rank-one terms with opposite signs in the canonical tensor become unbounded in size as the canonical tensor approaches the target tensor, in a phenomenon called degeneracy [9]. On the other hand, exact CP decomposition has been shown to be unique up to scaling and permutation under mild conditions that relate the ranks of the factor matrices with the tensor rank [9].

A multitude of algorithms and approaches exist for computing approximate CP decompositions, see, for example, [17, 9, 1] and references therein. A standard numerical method for computing the CP approximation of Optimization Problem I is the ALS method, which was already proposed in early papers on CP decomposition [3, 7]. ALS is simple to understand and implement, and often performs adequately, but its convergence can also be very slow, depending on the problem and the initial condition. Alternatives to ALS are described in, for example, [17, 4, 5], see also the discussion in [9, 1]. Even though ALS is a simple algorithm, it has proven difficult over the years to develop alternatives that significantly improve on it in a way that is robust over large classes of problems. As a results, ALS-type algorithms are still often considered as the ‘workhorse’ algorithms of choice in practice.

In recent work on CP decomposition, Acar et al. [1] consider first-order optimization algorithms for Optimization Problem I. In particular, they employ the nonlinear conjugate gradient (N-CG) method and compare it with ALS and nonlinear least-squares algorithms. In order to formulate first-order optimization methods, they derive expressions for the gradient of objective function (1.2) in a systematic way. At any (local) minimum of Optimization Problem I the following conditions hold:

FIRST-ORDER OPTIMALITY EQUATIONS I:

$$\nabla f(\mathcal{A}_R) = \mathbf{g}(\mathcal{A}_R) = 0. \quad (1.3)$$

Detailed expressions for the gradient vector  $\mathbf{g}(\mathcal{A}_R)$  will be given below. Acar et al. [1] then apply a standard N-CG optimization method with line search and obtain convergence speeds that are competitive with ALS and sometimes better, depending on the problem.

In this paper we also follow a first-order optimization approach for Optimization Problem I, but we propose to use a different optimization method. In particular, we propose to use a nonlinear GMRES approach to recombine iterates provided by a standard iterative method for Optimization Problem I (we use ALS as the ‘GMRES preconditioner’ in this paper), combined with line search for globalization. The resulting N-GMRES optimization algorithm will be shown numerically to accelerate convergence significantly in many cases, especially when stationary points need to be determined accurately for difficult problems, and we show this for two classes of dense and sparse test problems. The N-GMRES optimization algorithm proposed is easy to implement as a wrapper around any iterative solution method for Optimization Problem I. It combines previous iterates and can thus, in that sense, also be considered as a generalization of line search methods for ALS [17, 14], but taking more previous iterates into account (we typically use up to 20). However, our approach is also more general in that it can potentially be used to accelerate other CP algorithms than ALS. In fact, the N-GMRES optimization algorithm proposed is based on general concepts and may be applied to other nonlinear optimization problems, as a potential alternative to other first-order optimization methods like N-CG.

The nonlinear GMRES acceleration method we propose to use in our algorithm for nonlinear optimization is the nonlinear extension of GMRES that was developed by Washio and Oosterlee in [18] to accelerate multigrid solvers for nonlinear PDE systems (see also [13, 12] for further applications of their method in the nonlinear PDE systems context). Their method is a generalization of Saad and Schultz’s celebrated GMRES method for linear equation systems [15] to the nonlinear case. In the N-GMRES optimization algorithm we propose, we apply this nonlinear GMRES approach to combine ALS-generated iterates with the goal of driving the residual of the

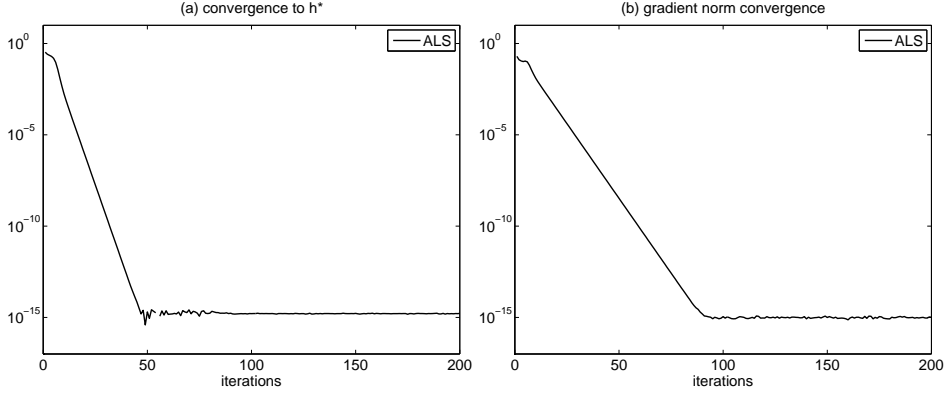


FIG. 1.1. *ALS convergence plots for an ‘easy’ instance of Test Problem I (parameters  $s=50$ ,  $c=0.5$ ,  $R=3$ ,  $l_1=1$ ,  $l_2=1$ , see Section 3 for full problem description). (a) Convergence of  $|h(\mathcal{A}_R^{(i)}) - h^*|$ , where  $h^*$  is the value of the normalized distance measure, (1.4), in the stationary point the method converges to. (b) Convergence of the normalized gradient of the objective function,  $\|\mathbf{g}(\mathcal{A}_R^{(i)})\|_2 / \|\mathcal{T}\|_F$ , indicating convergence to a stationary point. ALS is a fast method for this ‘easy’ problem, and N-GMRES acceleration is not required.*

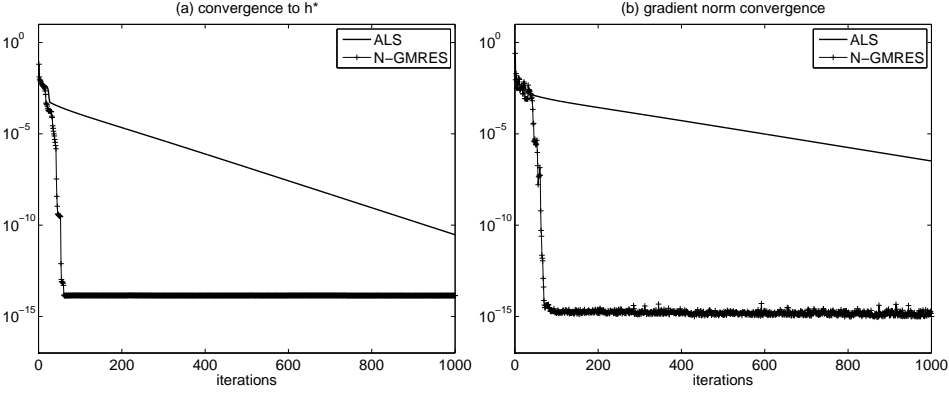


FIG. 1.2. *ALS convergence plots for a ‘difficult’ instance of Test Problem I (parameters  $s=50$ ,  $c=0.9$ ,  $R=3$ ,  $l_1=1$ ,  $l_2=1$ ). (a) Convergence of  $|h(\mathcal{A}_R^{(i)}) - h^*|$ . (b) Convergence of  $\|\mathbf{g}(\mathcal{A}_R^{(i)})\|_2 / \|\mathcal{T}\|_F$ . ALS is slow for this ‘difficult’ problem, but our proposed N-GMRES optimization algorithm significantly reduces the number of iterations.*

(nonlinear) first-order optimality equations to zero. A line search method is used to provide globalization. Just like for GMRES for linear equation systems, the method that generates the iterates can be seen as a preconditioner for GMRES, or, from an alternative viewpoint, GMRES can be interpreted as an acceleration mechanism for the method that generates the iterates [18, 13]. It will be explained in detail below how this interpretation carries over to the present context of nonlinear optimization for CP decomposition.

Figs. 1.1-1.3 give a preview of what the N-GMRES optimization algorithm proposed in this paper tries to achieve. The figures present convergence plots for a dense test problem that is a standard test case for CP decomposition from the literature [17, 1]. A 3-way data tensor with size  $50 \times 50 \times 50$  is generated starting from a

rank-3 canonical tensor with random factor matrices modified to have pre-specified collinearity  $c$ , and noise is added. (See Test Problem I in Section 3 for a detailed description of the test case.). A rank-3 CP approximation is sought starting from a random initial guess. In order to track accuracy during the progress of the iterative methods, we define a measure of the normalized distance between data tensor  $\mathcal{T}$  and rank- $R$  approximation  $\mathcal{A}_R^{(i)}$  in iteration  $i$  as

$$h(\mathcal{A}_R^{(i)}) = \frac{\|\mathcal{T} - \mathcal{A}_R^{(i)}\|_F}{\|\mathcal{T}\|_F}, \quad (1.4)$$

and define the optimal distance

$$h^* = h(\mathcal{A}_R^*), \quad (1.5)$$

where  $\mathcal{A}_R^*$  is the stationary point that the method converges to in the test. The accuracy of the approximation as the iterative method progresses is then tracked by measuring  $|h(\mathcal{A}_R^{(i)}) - h^*|$ . We also track convergence of  $\|\mathbf{g}(\mathcal{A}_R^{(i)})\|_2 / \|\mathcal{T}\|_F$ , the norm of the gradient of the objective function normalized by the norm of the data tensor, which gives information about convergence to a stationary point.

Fig. 1.1 shows convergence plots for a case with collinearity  $c = 0.5$ . It is known that this case is ‘easy’ for ALS, and the plots confirm that ALS converges quickly to a stationary point. It is also known that factor matrices with nearly collinear columns constitute problems that are more difficult for ALS [17, 1]. Fig. 1.2 shows convergence plots for a case with  $c = 0.9$ , and we can indeed observe that ALS converges slowly. The plots also show how the N-GMRES optimization algorithm that is proposed in this paper significantly speeds up ALS and reduces the number of iterations dramatically. Of course, Fig. 1.2 is only part of the story, because the N-GMRES iterations are more expensive than simple ALS iterations. However, the timing plots in Fig. 1.3 show that significant gains can still be made if this extra cost is taken into account, especially when accurate results are desired.

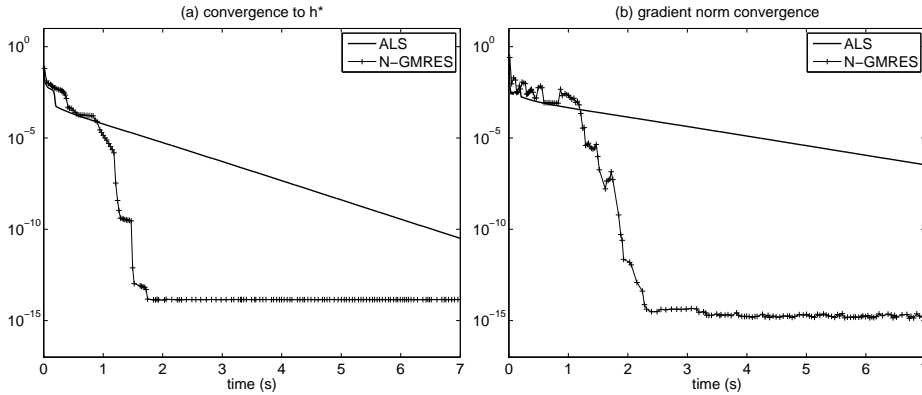


FIG. 1.3. Same as Fig. 1.2, but now convergence is plotted as a function of execution time. Even though N-GMRES iterations take more time per iteration, N-GMRES still substantially accelerates ALS, especially if accurate approximations are desired.

The rest of this paper is organized as follows. In Section 2 we describe the proposed N-GMRES optimization algorithm for nonlinear optimization problems, and

explain how it is applied to compute rank- $R$  canonical tensor decompositions. We also explain the interpretation of the N-GMRES acceleration method as a preconditioned GMRES nonlinear optimization method. In Section 3 we test the N-GMRES optimization algorithm on dense tensor test cases, and in Section 4 we discuss sparse test problems. Conclusions are formulated in Section 5.

**2. N-GMRES Optimization Algorithm for Canonical Tensor Decomposition.** In this section, we describe the proposed N-GMRES optimization algorithm for nonlinear optimization problems, and explain how it is applied to compute rank- $R$  canonical tensor decompositions. We start with a description of the N-GMRES optimization approach, and situate this discussion in the context of a general nonlinear optimization problem, since the approach is general. We then compare the N-GMRES approach for optimization to GMRES for linear systems, explaining how the dual viewpoint of preconditioned GMRES and GMRES acceleration applies in the nonlinear optimization context. In our descriptions we closely follow the derivations and presentation of Washio and Oosterlee's nonlinear GMRES for PDE problems from [18, 13], which we propose to use as the main ingredient in our nonlinear optimization algorithm. We give extensive details because these details give precise insight into how the GMRES method generalizes to the nonlinear optimization problem. This is followed by a discussion of how N-GMRES is applied to CP optimization, giving details on the first-order optimality equations for CP and the line search algorithm we use in the numerical results of subsequent sections.

**2.1. N-GMRES Optimization Algorithm.** Consider the following nonlinear optimization problem with associated first-order optimality equations:

OPTIMIZATION PROBLEM II:

$$\text{find } \mathbf{u}^* \text{ that minimizes } f(\mathbf{u}). \quad (2.1)$$

FIRST-ORDER OPTIMALITY EQUATIONS II:

$$\nabla f(\mathbf{u}) = \mathbf{g}(\mathbf{u}) = 0. \quad (2.2)$$

Assume that we have  $i + 1$  previous iterates  $\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_{i-1}, \mathbf{u}_i$ , and that we have a one-step (nonlinear) iterative update process  $M(\cdot)$  that generates a new approximation from an existing approximation:

$$\mathbf{u}_{new} = M(\mathbf{u}_{old}). \quad (2.3)$$

The goal of the N-GMRES optimization algorithm will be to accelerate the convergence of iterative update process  $M(\cdot)$ .

Every iteration of the N-GMRES optimization algorithm consists of three steps.

STEP I:

In the first step, we generate a preliminary new iterate  $\bar{\mathbf{u}}_{i+1}$  from the last iterate  $\mathbf{u}_i$  using one-step iterative update process  $M(\cdot)$ :

$$\bar{\mathbf{u}}_{i+1} = M(\mathbf{u}_i). \quad (2.4)$$

STEP II:

In the second step, we find an accelerated iterate  $\hat{\mathbf{u}}_{i+1}$  that is obtained by recombining previous iterates as follows:

$$\hat{\mathbf{u}}_{i+1} = \bar{\mathbf{u}}_{i+1} + \sum_{j=0}^i \alpha_j (\bar{\mathbf{u}}_{i+1} - \mathbf{u}_j). \quad (2.5)$$

The N-GMRES algorithm seeks to determine the unknown coefficients  $\alpha_j$  such that the two-norm of the gradient of the objective function evaluated at the accelerated iterate is small. We call  $\mathbf{g}(\hat{\mathbf{u}}_{i+1})$  the residual at  $\hat{\mathbf{u}}_{i+1}$ , and the objective is thus to determine the  $\alpha_j$  such that the residual is minimized in the two-norm: we attempt to minimize  $\|\mathbf{g}(\hat{\mathbf{u}}_{i+1})\|_2$ . However,  $\mathbf{g}(\cdot)$  is a nonlinear function of the  $\alpha_j$  (more precisely, it is a high-order polynomial in the  $\alpha_j$ ), and we proceed by linearization to allow for inexpensive computation of coefficients  $\alpha_j$  that may approximately minimize  $\|\mathbf{g}(\hat{\mathbf{u}}_{i+1})\|_2$ . Using the following approximations

$$\begin{aligned} \mathbf{g}(\hat{\mathbf{u}}_{i+1}) &\approx \mathbf{g}(\bar{\mathbf{u}}_{i+1}) + \sum_{j=0}^i \left. \frac{\partial \mathbf{g}}{\partial \mathbf{u}} \right|_{\bar{\mathbf{u}}_{i+1}} \alpha_j (\bar{\mathbf{u}}_{i+1} - \mathbf{u}_j) \\ &\approx \mathbf{g}(\bar{\mathbf{u}}_{i+1}) + \sum_{j=0}^i \alpha_j (\mathbf{g}(\bar{\mathbf{u}}_{i+1}) - \mathbf{g}(\mathbf{u}_j)) \end{aligned} \quad (2.6)$$

we arrive at minimization problem

find coefficients  $(\alpha_0, \dots, \alpha_i)$  that minimize

$$\|\mathbf{g}(\bar{\mathbf{u}}_{i+1}) + \sum_{j=0}^i \alpha_j (\mathbf{g}(\bar{\mathbf{u}}_{i+1}) - \mathbf{g}(\mathbf{u}_j))\|_2.$$

This is a standard least-squares problem that can be solved, for example, by using the normal equations. Defining

$$\begin{aligned} \boldsymbol{\alpha} &= (\alpha_0, \dots, \alpha_i)^T, \\ \mathbf{p}_j &= \mathbf{g}(\bar{\mathbf{u}}_{i+1}) - \mathbf{g}(\mathbf{u}_j), \\ \mathbf{P} &= [\mathbf{p}_0 | \dots | \mathbf{p}_i], \end{aligned}$$

we minimize  $\|\mathbf{P}\boldsymbol{\alpha} + \mathbf{g}(\bar{\mathbf{u}}_{i+1})\|_2$ , which leads to normal equation system

$$\mathbf{P}^T \mathbf{P} \boldsymbol{\alpha} = -\mathbf{P}^T \mathbf{g}(\bar{\mathbf{u}}_{i+1}). \quad (2.7)$$

STEP III:

In the third step, we perform a line search that optimizes objective function  $\mathbf{f}(\mathbf{u})$  on a half line starting at preliminary iterate  $\bar{\mathbf{u}}_{i+1}$ , which was generated in Step I, and connecting it with accelerated iterate  $\hat{\mathbf{u}}_{i+1}$ , which was generated in Step II:

$$\text{find } \beta^* \in [0, \infty) \text{ that minimizes} \quad (2.8)$$

$$\mathbf{f}(\bar{\mathbf{u}}_{i+1} + \beta(\hat{\mathbf{u}}_{i+1} - \bar{\mathbf{u}}_{i+1})). \quad (2.9)$$

This line search step is necessary, because without it erratic convergence behavior can occur, especially as long as iterates are far from a stationary point. In this case, the linearization steps in (2.6) may incur large errors, resulting in accelerated iterates

$\hat{\mathbf{u}}_{i+1}$  that may be bad approximations. The result of the line search is finally selected as the new iterate  $\mathbf{u}_{i+1}$ :

$$\mathbf{u}_{i+1} = \bar{\mathbf{u}}_{i+1} + \beta^*(\hat{\mathbf{u}}_{i+1} - \bar{\mathbf{u}}_{i+1}). \quad (2.10)$$

In practice, to limit the computational cost and especially the memory cost of the N-GMRES acceleration, we implement the algorithm with a windowed acceleration process with window size  $w$ , in which N-GMRES-accelerated iterates are generated based on the  $w$  last iterations.

Fig. 2.1 gives a schematic representation of the N-GMRES optimization algorithm, and Algorithm 1 describes it in pseudo-code. (Note of course that the  $w$  initial iterates required in the pseudo-code description can naturally be generated by applying the algorithm with a window size that gradually increases from one up to  $w$ , starting from a single initial guess.)

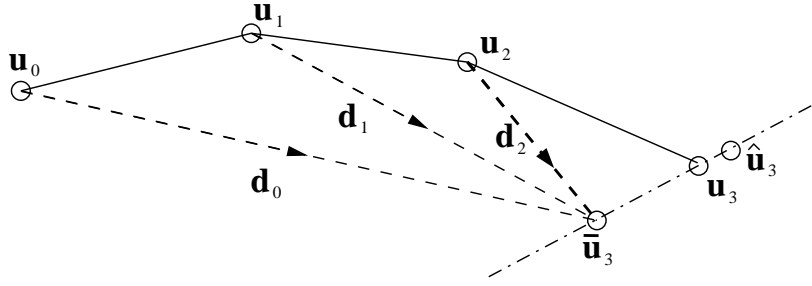


FIG. 2.1. Schematic representation of one iteration of the N-GMRES optimization algorithm. Given previous iterations  $\mathbf{u}_0$ ,  $\mathbf{u}_1$  and  $\mathbf{u}_2$ , new iterate  $\mathbf{u}_3$  is generated as follows. In Step I, preliminary iterate  $\bar{\mathbf{u}}_3$  is generated by the one-step update process  $M(\cdot)$ :  $\bar{\mathbf{u}}_3 = M(\mathbf{u}_2)$ . In Step II, the nonlinear GMRES step, accelerated iterate  $\hat{\mathbf{u}}_3$  is obtained by determining the coefficients  $\alpha_j$  in  $\hat{\mathbf{u}}_3 = \bar{\mathbf{u}}_3 + \alpha_0 \mathbf{d}_0 + \alpha_1 \mathbf{d}_1 + \alpha_2 \mathbf{d}_2$  such that the gradient of the objective function in  $\hat{\mathbf{u}}_3$  is approximately minimized. In Step III, the new iterate,  $\mathbf{u}_3$ , is finally generated by a line search that minimizes the objective function  $\mathbf{f}(\bar{\mathbf{u}}_3 + \beta(\hat{\mathbf{u}}_3 - \bar{\mathbf{u}}_3))$ .

---

**Algorithm 1:** N-GMRES optimization algorithm (window size  $w$ )

---

**Input:**  $w$  initial iterates  $\mathbf{u}_0, \dots, \mathbf{u}_{w-1}$ .

---

$i = w - 1$

**repeat**

STEP I: (generate preliminary iterate by one-step update process  $M(\cdot)$ )

$$\bar{\mathbf{u}}_{i+1} = M(\mathbf{u}_i)$$

STEP II: (generate accelerated iterate by nonlinear GMRES step)

$$\hat{\mathbf{u}}_{i+1} = \text{gmres}(\mathbf{u}_{i-w+1}, \dots, \mathbf{u}_i; \bar{\mathbf{u}}_{i+1})$$

STEP III: (generate new iterate by line search process)

$$\mathbf{u}_{i+1} = \text{linesearch}(\bar{\mathbf{u}}_{i+1} + \beta(\hat{\mathbf{u}}_{i+1} - \bar{\mathbf{u}}_{i+1}))$$

$i = i + 1$

**until** convergence criterion satisfied

---

We now give some additional remarks about the N-GMRES optimization algorithm proposed and its application to canonical tensor decomposition.

First, for Step I in the algorithm, we use the ALS algorithm as the one-step update process  $M(\cdot)$ . The ALS algorithm for CP decomposition will be described in



Section 2.3, along with specific expressions for the gradient of the objective function and the first-order optimality equations of Optimization Problem I. The one-step update process  $M(\cdot)$  can be interpreted as the preconditioner of the nonlinear GMRES procedure, as will be explained in Section 2.2.

Second, Step I and Step II in the N-GMRES optimization algorithm are entirely analogous to the nonlinear GMRES approach for PDE systems that was proposed by Washio and Oosterlee in [18]. In their applications, they employ nonlinear GMRES to drive the residuals of the PDEs to zero, and use multigrid as the preconditioner [18, 13, 12]. In the present context of canonical tensor decomposition as a nonlinear optimization problem, we drive the residual of the first-order optimality equations (the gradient of the objective function) to zero, and use ALS as the preconditioner.

Third, Step III in the N-GMRES optimization algorithm performs a role analogous to the selection mechanism proposed by Washio and Oosterlee in [18] to reduce erratic convergence. They select either the preliminary iterate or the accelerated one, based on ingenious but somewhat ad-hoc criteria that consider changes in the norms of the residuals and in the solution updates. Instead, in the present context of nonlinear optimization problems, we can exploit the final goal of minimizing the objective function to control erratic behavior, and rather than selecting either the preliminary iterate or the accelerated one, we perform a line search on the line connecting the two. Not only does this control erratic behavior, but it also provides an improved new iterate, at the cost of a line search.

Fourth, it may be beneficial to restart the N-GMRES acceleration with window size one when indications arise that the current window contains iterates that harm the acceleration process. In [18] the GMRES procedure is restarted based on criteria similar to the ones that are used to choose between the preliminary or accelerated iterate. In the context of N-GMRES for nonlinear optimization problems, we propose the following simple criterion: we restart whenever the search direction of the line search,  $\hat{\mathbf{u}}_{i+1} - \bar{\mathbf{u}}_{i+1}$ , does not point in a descent direction (so we set the window size back to one whenever  $\mathbf{g}(\bar{\mathbf{u}}_{i+1})^T (\hat{\mathbf{u}}_{i+1} - \bar{\mathbf{u}}_{i+1}) > 0$ ). The motivation for this is simple: we expect the acceleration mechanism to be effective close to a stationary point. In that case, the accelerated iterate is expected to be an improvement over the preliminary iterate, and the search direction is expected to be a descent direction. If this is not the case, this is taken as an indication that the current N-GMRES sequence does not help, and N-GMRES is restarted. Our numerical tests have indicated that this simple approach indeed tends to be beneficial for speed of convergence, for the test problems we considered. Note that windowing with restarts can be implemented efficiently in an elegant way, and we refer to the detailed pseudocode in [18] for this.

Fifth, another practical point is that the normal equation system in (2.7) may become ill-conditioned since the vectors  $\mathbf{p}_j$  may become nearly linearly dependent. One way to deal with this, as suggested in [18], is to add a small multiple of the identity matrix,  $\delta \mathbf{I}$ , to the normal equation operator. This was sufficient for the numerical tests we performed, and [18] provides further theoretical justification for this fix.

Sixth, for the line search of Step III, we use in our numerical experiments the Moré-Thuente line search method [10] as implemented in the Poblano toolbox for Matlab [6]. This line search method was also used in the N-CG method for canonical tensor decomposition in [1]. This is a rather sophisticated line search method that employs multiple function and gradient evaluations to improve the solution.

Finally, we briefly discuss the computational cost of the N-GMRES optimization

algorithm. In terms of memory cost, for window size  $w$  the algorithm requires storing  $w$  previous solution vectors and residual vectors. In the numerical results presented below we will see that, for the CP decomposition test problems we consider, a window size of approximately  $w = 20$  is optimal in terms of computing time. This requires substantial additional memory, however, and memory can obviously be traded for speed. In terms of the computing time, the dominant costs in each iteration of N-GMRES nonlinear optimization algorithm 1 are applying  $M(\cdot)$  in Step I, computing the gradient of the preliminary iterate,  $\mathbf{g}(\bar{\mathbf{u}}_{i+1})$ , in Step II, and the function and gradient evaluations in the line search of Step III. Since the optimization problems we solve in this paper are quite involved and the ALS, function and gradient evaluations are very expensive, these calculations strongly dominate all others in practice. The relevant extra costs of the N-GMRES optimization approach compared to just applying the one-step update process  $M(\cdot)$  are thus the additional function and gradient evaluations, and the cost of building and solving the normal equation system (2.7) and computing the accelerated iterate is negligible in practice. The number of function and gradient evaluations per line search depend on the line search method used and the accuracy parameters one chooses for the line search calculations, which in turn influence the number of overall iterations required to achieve a certain accuracy for the stationary point. It is thus difficult to say much in general about the additional cost of the line search step. The numerical results to be presented below show that, for two relevant classes of test problems for canonical tensor decomposition, N-GMRES optimization can be significantly faster than stand-alone iteration by applying  $M(\cdot)$ , especially when high accuracy is desired. We will at this point just give one example. For the N-GMRES CP calculation shown in Fig. 1.3, up to an accuracy of  $|h - h^*| < 10^{-10}$  (which required 54 iterations), about 40% of the time is spent in the ALS iterations (54 calls), 15% in the calculation of  $\mathbf{g}(\bar{\mathbf{u}}_{i+1})$  (54 calls), and about 30% in the function and gradient evaluations in the line search procedure (137 calls to evaluate  $\mathbf{f}$  and  $\mathbf{g}$ ). The average number of function/gradient evaluations per line search was about 2.5. While the generality of this single example is of course limited, it does illustrate that N-GMRES optimization algorithm 1 mainly adds overhead in terms of additional function and residual evaluations for the GMRES and the line search steps. But it is precisely these additional calculations that potentially lead to a great reduction in the number of iterations and overall execution time, as we will illustrate with extensive numerical tests below.

**2.2. Interpretation as a GMRES Method: Acceleration and Preconditioning.** In this subsection we briefly recall some particulars of the GMRES method for linear equation system

$$\mathbf{A} \mathbf{u} = \mathbf{b}, \quad (2.11)$$

and give a detailed exposition of the parallels with the N-GMRES optimization algorithm proposed in the previous subsection. We follow the presentation of [18], but explain in detail how the parallels apply in the nonlinear optimization context. As in [18], we discuss GMRES for (2.11) in a particular, perhaps non-standard way, which will allow us to draw parallels to the nonlinear optimization algorithm.

Our starting point for explaining the principles of preconditioned GMRES for linear equation system (2.11) is to consider so-called stationary iterative methods for (2.11) of the following form:

$$\mathbf{u}_{i+1} = \mathbf{u}_i + \mathbf{M}^{-1} \mathbf{r}_i. \quad (2.12)$$

Here,  $\mathbf{r}_i$  is the  $i$ th residual, defined by

$$\mathbf{r}_i = \mathbf{b} - \mathbf{A} \mathbf{u}_i, \quad (2.13)$$

and matrix  $\mathbf{M}$  is an approximation of  $\mathbf{A}$  that has an easily computable inverse, i.e.,  $\mathbf{M}^{-1} \approx \mathbf{A}^{-1}$ . For example,  $\mathbf{M}$  can be chosen to correspond to Gauss-Seidel or Jacobi iteration, or to a multigrid cycle [18]. General form (2.12) of a stationary iterative method can be motivated as follows. Defining the error of the  $i$ th iterate as  $\mathbf{e}_i = \mathbf{u} - \mathbf{u}_i$  (with  $\mathbf{u}$  the exact solution of (2.11)), it is easy to see that  $\mathbf{A} \mathbf{e}_i = \mathbf{r}_i$ . Observing that  $\mathbf{u} = \mathbf{u}_i + \mathbf{e}_i = \mathbf{u}_i + \mathbf{A}^{-1} \mathbf{r}_i$ , and that knowledge of  $\mathbf{A}^{-1}$  would (obviously) lead to the exact solution in one step, one can conclude that, if one does not know  $\mathbf{A}^{-1}$  but does have access to an easily computable approximation  $\mathbf{M}^{-1}$ , update formula (2.12) may lead to a useful iteration process.

Consider a sequence of iterates  $\mathbf{u}_0, \dots, \mathbf{u}_i$  generated by update formula (2.12), starting from some initial guess  $\mathbf{u}_0$ . Note that the residuals of these iterates are related as follows:

$$\begin{aligned} \mathbf{r}_i &= \mathbf{b} - \mathbf{A} \mathbf{u}_i \\ &= (\mathbf{I} - \mathbf{A} \mathbf{M}^{-1}) \mathbf{r}_{i-1} \\ &= (\mathbf{I} - \mathbf{A} \mathbf{M}^{-1})^i \mathbf{r}_0. \end{aligned} \quad (2.14)$$

This motivates the definition of the following vector spaces:

$$\begin{aligned} V_{1,i+1} &= \text{span}\{\mathbf{r}_0, \dots, \mathbf{r}_i\}, \\ V_{2,i+1} &= \text{span}\{\mathbf{r}_0, \mathbf{A} \mathbf{M}^{-1} \mathbf{r}_0, (\mathbf{A} \mathbf{M}^{-1})^2 \mathbf{r}_0, \dots, (\mathbf{A} \mathbf{M}^{-1})^i \mathbf{r}_0\} \\ &= K_{i+1}(\mathbf{A} \mathbf{M}^{-1}, \mathbf{r}_0), \end{aligned} \quad (2.15)$$

$$\begin{aligned} V_{3,i+1} &= \text{span}\{\mathbf{M}(\mathbf{u}_1 - \mathbf{u}_0), \mathbf{M}(\mathbf{u}_2 - \mathbf{u}_1), \dots, \mathbf{M}(\mathbf{u}_{i+1} - \mathbf{u}_i)\}, \\ V_{4,i+1} &= \text{span}\{\mathbf{M}(\mathbf{u}_{i+1} - \mathbf{u}_0), \mathbf{M}(\mathbf{u}_{i+1} - \mathbf{u}_1), \dots, \mathbf{M}(\mathbf{u}_{i+1} - \mathbf{u}_i)\}. \end{aligned} \quad (2.16)$$

Vector space  $V_{2,i+1}$  is the so-called Krylov space  $K_{i+1}(\mathbf{A} \mathbf{M}^{-1}, \mathbf{r}_0)$  of order  $i+1$ , generated by matrix  $\mathbf{A} \mathbf{M}^{-1}$  and vector  $\mathbf{r}_0$ . It is easy to see that the vector spaces defined above are equal [18]:

LEMMA 2.1.  $V_{1,i+1} = V_{2,i+1} = V_{3,i+1} = V_{4,i+1}$ .

*Proof.* First,  $V_{1,i+1} = V_{2,i+1}$  by (2.14), which directly shows that  $\mathbf{r}_j \in V_{2,i+1}$  for all  $j$ , and  $(\mathbf{A} \mathbf{M}^{-1})^j \mathbf{r}_0 \in V_{1,i+1}$  for all  $j$  follows by recursion.

Second,  $V_{2,i+1} = V_{3,i+1}$  because  $\mathbf{M}(\mathbf{u}_{i+1} - \mathbf{u}_i) = \mathbf{r}_i$ , by (2.12).

Third,  $V_{3,i+1} = V_{4,i+1}$  because, for  $k < i+1$ ,  $\mathbf{u}_{i+1} - \mathbf{u}_k = \sum_{j=k+1}^{i+1} (\mathbf{u}_j - \mathbf{u}_{j-1})$ , and  $\mathbf{u}_k - \mathbf{u}_{k-1} = (\mathbf{u}_{i+1} - \mathbf{u}_{k-1}) - (\mathbf{u}_{i+1} - \mathbf{u}_k)$ .  $\square$

We know that  $\mathbf{M}(\mathbf{u}_{i+1} - \mathbf{u}_i) \in K_{i+1}(\mathbf{A} \mathbf{M}^{-1}, \mathbf{r}_0)$ . The GMRES procedure can be seen as a way to accelerate stationary iterative method (2.12), by recombining iterates (or, equivalently, by reusing residuals). In particular, we seek a better approximation  $\hat{\mathbf{u}}_{i+1}$ , with  $\mathbf{M}(\hat{\mathbf{u}}_{i+1} - \mathbf{u}_i)$  in the Krylov space  $K_{i+1}(\mathbf{A} \mathbf{M}^{-1}, \mathbf{r}_0)$ , such that  $\hat{\mathbf{r}}_{i+1} = \mathbf{b} - \mathbf{A} \hat{\mathbf{u}}_{i+1}$  has minimal two-norm. In other words, we seek optimal coefficients  $\beta_j$  in

$$\mathbf{M}(\hat{\mathbf{u}}_{i+1} - \mathbf{u}_i) = \sum_{j=0}^i \beta_j \mathbf{M}(\mathbf{u}_{i+1} - \mathbf{u}_j),$$

or in

$$\begin{aligned}\hat{\mathbf{u}}_{i+1} &= \mathbf{u}_i + \sum_{j=0}^i \beta_j (\mathbf{u}_{i+1} - \mathbf{u}_j) \\ &= \mathbf{u}_{i+1} - (\mathbf{u}_{i+1} - \mathbf{u}_i) + \sum_{j=0}^i \beta_j (\mathbf{u}_{i+1} - \mathbf{u}_j).\end{aligned}$$

Equivalently, we can seek optimal coefficients  $\alpha_j$  in

$$\hat{\mathbf{u}}_{i+1} = \mathbf{u}_{i+1} + \sum_{j=0}^i \alpha_j (\mathbf{u}_{i+1} - \mathbf{u}_j), \quad (2.17)$$

such that  $\|\hat{\mathbf{r}}_{i+1}\|_2$  is minimized (which leads to a small least-squares problem). Apart from the power of the minimization principle, GMRES for linear systems is especially powerful because this minimization can be done very efficiently, using the Arnoldi iteration to generate orthogonal bases for Krylov spaces of increasing order [15]. For nonlinear problems, such an advantageous implementation is not possible, but the general ideas of optimal acceleration remain powerful and can still be applied.

We now explain two complementary viewpoints of GMRES for linear system (2.11). We have presented the method as a way to accelerate simple one-step stationary iterative method (2.12). A more customary way to see GMRES is in terms of preconditioning. With  $\mathbf{M} = \mathbf{I}$ , the approach described above reduces to ‘non-preconditioned’ GMRES. Preconditioned GMRES can then also be derived by applying non-preconditioned GMRES to the preconditioned linear equation system  $\mathbf{A}\mathbf{M}^{-1}(\mathbf{M}\mathbf{u}) = \mathbf{b}$ . In this viewpoint, the matrix  $\mathbf{M}^{-1}$  is called the preconditioner matrix, because its role is viewed as to pre-condition the spectrum of the linear system operator such that the (non-preconditioned) GMRES method applied to  $(\mathbf{A}\mathbf{M}^{-1})\mathbf{y} = \mathbf{b}$  becomes more effective. By extension, it is also customary to say that the stationary iterative method preconditions GMRES (in the sense of, for example, Gauss-Seidel or multigrid being used as preconditioners for GMRES). In this viewpoint, the role of the stationary iterative method is to generate a preconditioned residual that builds the Krylov space.

The stage is now set for discussing the interpretation of the N-GMRES optimization algorithm of the previous subsection as a preconditioned GMRES method (along the lines of [18] for nonlinear GMRES applied to nonlinear PDE systems). Before we do so, we need to add one important remark regarding GMRES for linear systems. In the presentation above, all iterates  $\mathbf{u}_j$  for  $j = 0, \dots, i$  (for instance, in the right-hand side of (2.17)) refer to iterates generated by stationary iterative method (2.12), without acceleration. However, the formulas remain valid when we use accelerated iterates instead; this merely changes the values of the coefficients  $\alpha_j$ , but leads to the same accelerated iterates [18]. The reason is that all the GMRES optimization does is to produce improved iterates with residuals that are optimal elements of the Krylov spaces, but the Krylov spaces are still the ones generated by the residuals of the stationary iterative method, and do themselves not change, due to linearity.

The N-GMRES optimization algorithm presented in Subsection 2.1 can be interpreted as a preconditioned GMRES method as follows. With  $\mathbf{u}_{i+1}$  the preliminary  $i + 1$ st iterate generated by (2.12) and the  $\mathbf{u}_j$  with  $j \leq i$  the previous accelerated iterates, the expression for the accelerated iterate  $\hat{\mathbf{u}}_{i+1}$  we seek in (2.17) for the linear

case corresponds directly to the expression in (2.5) for the nonlinear case. The definition of  $V_{2,i+1}$  in (2.15) does not provide a nonlinear generalization for the Krylov space in which we seek an optimal approximation in the linear case, but the definition of  $V_{4,i+1}$  (2.16) does: the image of the linear Krylov space  $K_{i+1}(\mathbf{A}\mathbf{M}^{-1}, \mathbf{r}_0)$  under the preconditioning matrix  $\mathbf{M}^{-1}$  is  $\text{span}\{(\mathbf{u}_{i+1} - \mathbf{u}_0), (\mathbf{u}_{i+1} - \mathbf{u}_1), \dots, (\mathbf{u}_{i+1} - \mathbf{u}_i)\}$ , and this can trivially be generalized to the nonlinear case: the nonlinear generalization of the Krylov space is precisely given by this vector space. And it is precisely in this space that we seek an optimal vector, both in the linear and in the nonlinear case (see (2.17) and (2.5), respectively). Stationary iterative method (2.12) is the preconditioning process for GMRES in the linear case, responsible for generating a preliminary approximation with the help of a residual calculation and a preconditioning matrix, and in the same way one-step approximation method (2.4) is the preconditioner for GMRES in the nonlinear case. In the present case of the N-GMRES optimization algorithm applied to canonical tensor decomposition, we can say that ALS is used as the preconditioner of N-GMRES. The alternative viewpoint is that N-GMRES accelerates ALS.

A variety of preconditioners exist for linear systems of equations, and research in this area has been a very active field ever since GMRES was proposed. In this paper, we accelerate ALS for the canonical tensor decomposition optimization problem, but the preconditioning interpretation of the N-GMRES optimization algorithm suggests that it may be worthwhile to explore different N-GMRES preconditioners for the canonical tensor decomposition optimization problem in future work. Similarly, it may be interesting to explore preconditioned N-GMRES algorithms for other nonlinear optimization problems. (Or, put in another way, there appears to be potential for N-GMRES to accelerate already existing iterative methods for other nonlinear optimization problems.)

**2.3. First-Order Optimality Equations for the CP Optimization Problem.** In this subsection, we briefly recall the first-order optimality equations for CP Optimization Problem I, following [1]. The gradient of objective function (1.2) can be written as a vector of matrices

$$\nabla f(\mathcal{A}_R) = \vec{\mathbf{G}}(\mathcal{A}_R) = (\mathbf{G}^{(1)}, \dots, \mathbf{G}^{(N)}), \quad (2.18)$$

with  $\mathbf{G}^{(n)} \in \mathbb{R}^{I_n \times R}$  ( $n = 1, \dots, N$ ). The matrices  $\mathbf{G}^{(n)}$  for  $n = 1, \dots, N$  are given by

$$\mathbf{G}^{(n)} = -\mathbf{T}_{(n)} \bar{\Phi}^{(n)} + \mathbf{A}^{(n)} \bar{\Gamma}^{(n)}, \quad (2.19)$$

with variables defined as follows.

With  $\mathcal{T}$  and  $\mathcal{A}_R \in \mathbb{R}^{I_1 \times \dots \times I_N}$ , define size parameters

$$K = \prod_{l=1}^N I_l, \quad (2.20)$$

$$\bar{K}^{(n)} = \prod_{\substack{l=1 \\ l \neq n}}^N I_l. \quad (2.21)$$

Then  $\mathbf{T}_{(n)} \in \mathbb{R}^{I_n \times \bar{K}^{(n)}}$  is the mode- $n$  matricization of  $\mathcal{T}$ , obtained by stacking the  $n$ -mode fibres of  $\mathcal{T}$  in its columns in a regular way, see [9]. Similar to size parameters

$K$  and  $\bar{K}^{(n)}$ , we define matrices  $\Phi \in \mathbb{R}^{K \times R}$  and  $\bar{\Phi}^{(n)} \in \mathbb{R}^{\bar{K}^{(n)} \times R}$  by

$$\Phi = \mathbf{A}^{(1)} \odot \dots \odot \mathbf{A}^{(N)}, \quad (2.22)$$

$$\bar{\Phi}^{(n)} = \mathbf{A}^{(1)} \odot \dots \odot \mathbf{A}^{(n-1)} \odot \mathbf{A}^{(n+1)} \odot \dots \odot \mathbf{A}^{(N)}, \quad (2.23)$$

where  $\odot$  is the Khatri-Rao product [9]. Finally, the matrices  $\Gamma \in \mathbb{R}^{R \times R}$  and  $\bar{\Gamma}^{(n)} \in \mathbb{R}^{R \times R}$  are defined by

$$\Gamma = (\mathbf{A}^{(1)T} \mathbf{A}^{(1)}) * \dots * (\mathbf{A}^{(N)T} \mathbf{A}^{(N)}), \quad (2.24)$$

$$\begin{aligned} \bar{\Gamma}^{(n)} &= (\mathbf{A}^{(1)T} \mathbf{A}^{(1)}) * \dots * (\mathbf{A}^{(n-1)T} \mathbf{A}^{(n-1)}) * \\ &\quad (\mathbf{A}^{(n+1)T} \mathbf{A}^{(n+1)}) * \dots * (\mathbf{A}^{(N)T} \mathbf{A}^{(N)}), \end{aligned} \quad (2.25)$$

where  $*$  means element-wise multiplication.

The first-order optimality equations are then given by

$$\mathbf{G}^{(n)} = 0 = -\mathbf{T}_{(n)} \bar{\Phi}^{(n)} + \mathbf{A}^{(n)} \bar{\Gamma}^{(n)}, \quad n = 1, \dots, N. \quad (2.26)$$

They offer an easy way to describe the ALS method for CP optimization: an ALS iteration proceeds by sequentially solving for each of the factor matrices  $\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)}$  using the corresponding optimality equation  $\mathbf{G}^{(n)} = 0$ , updating the matrices  $\bar{\Phi}^{(n)}$  and  $\bar{\Gamma}^{(n)}$  along the way. As such, it is an example of a nonlinear block Gauss-Seidel method to solve the nonlinear equation system provided by the first-order optimality equations. The Matlab Tensor Toolbox [2] implements efficient methods for computing the product  $\mathbf{T}_{(n)} \bar{\Phi}^{(n)}$  both for sparse and dense tensors.

One point that deserves special attention is the following: if no special care is taken during ALS iterations, it may happen that vectors in some modes diverge while others approach zero size, even if the process is converging to the desired CP decomposition; this is indeed possible due to the scaling indeterminacy. This type of anomalous behavior can be avoided by normalizing the columns of the factor matrices. After each complete ALS iteration, for each rank-one term we first normalize all factor vectors to size one, and then distribute the product of the normalization factors evenly to all factor vectors (using the  $n$ th root). We also order the rank-one terms in order of decreasing product of normalization factors. We apply this normalization and reordering after each ALS iteration, but also after each calculation of new accelerated or line search iterates in the N-GMRES optimization procedure. This controls possible erratic behavior in the order of the rank-one components and balances the relative sizes of the factor vectors as they are stacked in the iterate vectors  $\mathbf{u}_i$ , which is expected to be beneficial for the convergence of the N-GMRES acceleration process. This consistent normalization thus appears to take care of the scaling and permutation indeterminacies present in the CP optimization problem, at least for the problems we have tested, as our numerical results confirm.

**2.4. Application of N-GMRES to CP Optimization: Further Particulars and Parameter Choices for Numerical Tests.** In this subsection we discuss some further particulars of our application of N-GMRES to CP optimization, and discuss some parameters for the numerical results to be presented in the next two sections. As we mentioned before, and as in [1], we utilize the Moré-Thuente line search method [10] as implemented in the Poblano toolbox for Matlab [6]. For all experiments, the Moré-Thuente line search parameters used were as follows:  $10^{-4}$  for

the function value tolerance,  $10^{-2}$  for the gradient norm tolerance, a starting search step length of 1 and a maximum of 20 iterations. These values were also used for the N-CG tests in [1], and we use them for our N-CG comparison runs as well. We use the N-CG variant with Polak-Ribière update formula [11]. As suggested in [18], the parameter  $\delta$  in the term  $\delta \mathbf{I}$  added to normal equation matrix  $\mathbf{P}^T \mathbf{P}$  in (2.7), was chosen as  $\epsilon$  times the size of the largest diagonal element of  $\mathbf{P}^T \mathbf{P}$ , with  $\epsilon = 10^{-12}$ . We normally choose the N-GMRES window size  $w$  equal to 20, which is confirmed to be a good choice in numerical tests described below. All initial guesses are determined uniformly randomly, as in [14], and when we compare different methods they are given the same random initial guess. All numerical tests were run on a laptop with a dual-core 2.53 GHz Intel Core i5 processor and 4GB of 1067 MHz DDR3 memory. Matlab version 7.11.0.584 (R2010b) 64-bit (maci64) was used for all tests.

**3. Numerical Results: Dense Tensor Test Problem.** In this and the next section, we present extensive numerical tests for the N-GMRES optimization algorithm, compared with ALS and the N-CG algorithm of [1], accessed via the Matlab Tensor Toolbox [2]. In this section we present a class of dense test problems, and the next section deals with a sparse problem class.

Our dense test problem generates 3-way data tensors of various sizes starting from a canonical tensor with specified rank and random factor matrices that are modified to have pre-specified collinearity  $c$ , and noise is added. This is a standard CP test problem from [17], and was also used in [1]. The collinearity between different columns of factor matrix  $\mathbf{A}^{(n)}$  is defined as

$$c = \frac{\mathbf{a}_r^{(n)T} \mathbf{a}_s^{(n)}}{\|\mathbf{a}_r^{(n)}\|_2 \|\mathbf{a}_s^{(n)}\|_2}. \quad (3.1)$$

Collinearity close to 1 is known to lead to slow ALS convergence [17].

**TEST PROBLEM I.** Generate a 3-way data tensor  $\mathcal{T}$  of noise-free rank  $R$ , size  $s \times s \times s$ , collinearity  $c$ , and noise levels  $l_1$  and  $l_2$  as follows. First generate an  $R \times R$  matrix  $\mathbf{K}$  that has diagonal elements 1 and off-diagonal elements  $c$ , and compute the Cholesky factor  $\mathbf{C}$  of  $\mathbf{K}$ . Then generate  $n$  uniformly random  $s \times R$  matrices, orthonormalize their columns using the QR decomposition, and multiply on the right with  $\mathbf{C}$ . Then let  $\mathcal{T}_R$  be the canonical rank- $R$  tensor generated by these matrices as factor matrices. Two types of noise are added to  $\mathcal{T}_R$ . Generate tensors  $\mathcal{N}_1$  and  $\mathcal{N}_2 \in \mathbb{R}^{s \times s \times s}$  with elements drawn from the standard normal distribution. An intermediate tensor  $\hat{\mathcal{T}}$  is generated as  $\hat{\mathcal{T}} = \mathcal{T}_R + (100/l_1 - 1)^{-1/2} \|\mathcal{T}_R\|_F \mathcal{N}_1 / \|\mathcal{N}_1\|_F$ , and finally  $\mathcal{T}$  is obtained as  $\mathcal{T} = \hat{\mathcal{T}} + (100/l_2 - 1)^{-1/2} \|\hat{\mathcal{T}}\|_F (\mathcal{N}_2 * \hat{\mathcal{T}}) / \|\mathcal{N}_2 * \hat{\mathcal{T}}\|_F$ , where  $*$  denotes element-wise multiplication.

We perform a series of tests computing a rank- $R$  CP decomposition of the tensors  $\mathcal{T}$ , for various values of  $s, c, R, l_1$  and  $l_2$ . Note that in this paper we do not study so-called overfactoring effects [17, 1], in which numerical methods may produce approximations that do not give physically relevant information when the CP-rank  $R$  is chosen larger than some rank intrinsic to the data tensor. Our reason for not considering overfactoring is that in this paper we accelerate ALS by the N-GMRES optimization approach, and we have observed that we almost always converge to the same stationary point as ALS. Since the overfactoring properties of ALS have been studied extensively elsewhere [17, 1], we do not consider them here.



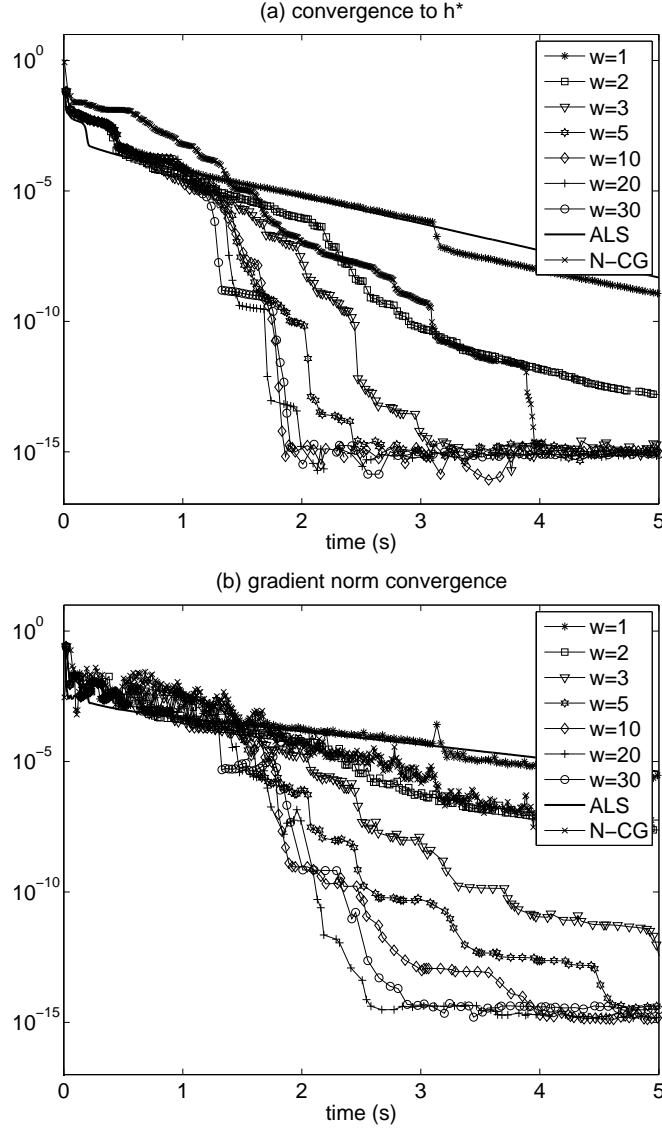


FIG. 3.1. Test Problem I with parameters ( $s = 50, c = 0.9, R = 3, l_1 = 1, l_2 = 1$ ). Convergence plots as a function of the N-GMRES window size,  $w$ . (a) Convergence of  $|h(\mathcal{A}_R^{(i)}) - h^*|$ , where  $h^*$  is the value of the normalized distance measure, (1.4), in the stationary point the method converges to. (b) Convergence of the normalized gradient of the objective function,  $\|\mathbf{g}(\mathcal{A}_R^{(i)})\|_2 / \|\mathcal{T}\|_F$ , indicating convergence to a stationary point. Window size  $w = 20$  emerges as a good choice for fast convergence when high accuracy is required.

Before performing tests for a wide range of parameter values  $s, c, R, l_1$  and  $l_2$ , we look at a specific case and study the choice of N-GMRES window size  $w$ . Fig. 3.1 shows convergence results for an instance of Test Problem I with parameters  $s = 50, c = 0.9, R = 3, l_1 = 1$  and  $l_2 = 1$ . This constitutes a ‘difficult’ case with  $c = 0.9$ , for which ALS converges rather slowly. It is the same case as in Figs. 1.2-1.3. Fig. 3.1 shows the effect of the window size  $w$  on convergence speed. Several observations can



$h^*$ accuracy $10^{-3}$		ALS		N-GMRES		N-CG	
problem parameters		it	time	it	time	it	time
1	$s=20, c=0.5, R=3, l_1=1, l_2=1$	18	<b>0.083</b>	16	0.21	34	0.17
2	$s=20, c=0.5, R=5, l_1=10, l_2=5$	9	<b>0.083</b>	8	0.17	64	0.51
3	$s=20, c=0.9, R=3, l_1=0, l_2=0$	186	0.8	153	1.7	137	<b>0.57</b>
4	$s=20, c=0.9, R=5, l_1=1, l_2=1$	19	<b>0.15</b>	13	0.34	195	1.4
5	$s=50, c=0.5, R=3, l_1=1, l_2=1$	11	<b>0.089</b>	8	0.21	38	0.46
6	$s=50, c=0.5, R=5, l_1=10, l_2=5$	10	<b>0.15</b>	9	0.3	50	0.97
7	$s=50, c=0.9, R=3, l_1=0, l_2=0$	314	2.2	56	<b>1.6</b>	200	1.8
8	$s=50, c=0.9, R=5, l_1=1, l_2=1$	15	<b>0.2</b>	10	0.43	>1821	>32
9	$s=100, c=0.5, R=3, l_1=1, l_2=1$	9	<b>0.31</b>	9	1.1	71	5.7
10	$s=100, c=0.5, R=5, l_1=10, l_2=5$	15	<b>0.68</b>	13	2.2	66	7.5
11	$s=100, c=0.9, R=3, l_1=0, l_2=0$	178	5.9	30	<b>3.9</b>	340	23
12	$s=100, c=0.9, R=5, l_1=1, l_2=1$	12	<b>0.52</b>	9	1.7	260	24

TABLE 3.1

Test Problem I. Number of iterations and time (in seconds) until accuracy measure  $|h(\mathcal{A}_R^{(i)}) - h^*|$  is reduced to  $10^{-3}$ . Here,  $h^*$  is the value of the normalized distance measure, (1.4), in the stationary point the methods converge to. The smallest times appear in bold. ALS is the fastest for most of these low-accuracy tests.

be made. The choice  $w = 1$  does not converge much faster than ALS, but  $w = 2$  already leads to significant gains when high accuracy is desired, and  $w = 3$  already performs reasonably similar to the best choice. (Note again that these convergence plots take into account the extra costs of the N-GMRES optimization and line search in each iteration.) The optimal performance appears to occur, for this test problem, when  $w \approx 20$  is chosen. Fast convergence of  $h(\mathcal{A}_R^{(i)})$  to the optimal value  $h^*$  (see (1.4)) is already fast obtained with high accuracy for  $w \approx 10$ , but Fig. 3.1(b) shows that  $w \approx 20$  leads to faster convergence to a stationary point with high accuracy. Convergence plots as a function of iteration (not shown due to space constraints) show the same pattern, indicating that speed differences are almost entirely due to differing iteration counts, and not to differences in the amount of work to build and solve normal equation system (2.7) for varying window size. We use window size  $w = 20$  in the numerical tests below.

Tables 3.1-3.3 show convergence results for a series of tests with a wide range of parameter values  $s, c, R, l_1$  and  $l_2$ . Table 3.1 compares the number of iterations and times (in seconds) that the ALS, N-GMRES and N-CG (from [1]) methods need to reduce the accuracy measure  $|h(\mathcal{A}_R^{(i)}) - h^*|$  to  $10^{-3}$ , where  $h^*$  is the value of the normalized distance measure, (1.4), in the stationary point the method converges to. All methods start from the same random initial guess, and converge to the same stationary point in these tests. This table compares the methods for a situation where low-accuracy results are considered sufficient. The smallest times appear in bold. Table 3.1 shows that ALS is normally the fastest when low accuracy is sufficient. This is consistent with what many others have observed for ALS-type algorithms, for CP decomposition and other problems, see, for example, [17, 1]. Note that the relative performance of the methods may be somewhat dependent on the initial guess, but this effect is partially averaged out by considering multiple tests with different parameters, and the results in the table give a representative overview of the relative performance of the methods.

$h^*$ accuracy $10^{-6}$		ALS		N-GMRES		N-CG	
problem parameters		it	time	it	time	it	time
1	$s=20, c=0.5, R=3, l_1=1, l_2=1$	25	<b>0.11</b>	19	0.25	42	0.2
2	$s=20, c=0.5, R=5, l_1=10, l_2=5$	21	<b>0.17</b>	13	0.28	77	0.58
3	$s=20, c=0.9, R=3, l_1=0, l_2=0$	949	4.1	167	1.9	197	<b>0.79</b>
4	$s=20, c=0.9, R=5, l_1=1, l_2=1$	582	4.2	109	<b>3.6</b>	614	3.9
5	$s=50, c=0.5, R=3, l_1=1, l_2=1$	20	<b>0.15</b>	11	0.29	50	0.56
6	$s=50, c=0.5, R=5, l_1=10, l_2=5$	20	<b>0.26</b>	14	0.53	67	1.3
7	$s=50, c=0.9, R=3, l_1=0, l_2=0$	1134	8	77	<b>2.4</b>	454	3.9
8	$s=50, c=0.9, R=5, l_1=1, l_2=1$	518	<b>5.9</b>	154	9.2	>1821	>32
9	$s=100, c=0.5, R=3, l_1=1, l_2=1$	19	<b>0.64</b>	12	1.4	98	7.3
10	$s=100, c=0.5, R=5, l_1=10, l_2=5$	27	<b>1.2</b>	16	2.9	115	11
11	$s=100, c=0.9, R=3, l_1=0, l_2=0$	>800	>27	69	<b>8.4</b>	673	44
12	$s=100, c=0.9, R=5, l_1=1, l_2=1$	457	19	85	<b>19</b>	620	52

TABLE 3.2

Test Problem I. Number of iterations and time (in seconds) until accuracy measure  $|h(\mathcal{A}_R^{(i)}) - h^*|$  is reduced to  $10^{-6}$ . The smallest times appear in bold. For these medium-accuracy tests, ALS is still fastest for all ‘easy’ cases ( $c = 0.5$ ), but N-GMRES or N-CG are faster for most of the ‘difficult’ cases ( $c = 0.9$ ).

$h^*$ accuracy $10^{-10}$		ALS		N-GMRES		N-CG	
problem parameters		it	time	it	time	it	time
1	$s=20, c=0.5, R=3, l_1=1, l_2=1$	37	<b>0.16</b>	22	0.3	52	0.24
2	$s=20, c=0.5, R=5, l_1=10, l_2=5$	37	<b>0.28</b>	17	0.39	97	0.7
3	$s=20, c=0.9, R=3, l_1=0, l_2=0$	>1600	>6.9	189	<b>2.4</b>	>400	>6.1
4	$s=20, c=0.9, R=5, l_1=1, l_2=1$	>1200	>8.6	139	<b>4.5</b>	1100	6.8
5	$s=50, c=0.5, R=3, l_1=1, l_2=1$	32	<b>0.23</b>	16	0.42	67	0.69
6	$s=50, c=0.5, R=5, l_1=10, l_2=5$	36	<b>0.44</b>	17	0.67	89	1.6
7	$s=50, c=0.9, R=3, l_1=0, l_2=0$	>1200	>8.5	104	<b>3.5</b>	>553	>7.6
8	$s=50, c=0.9, R=5, l_1=1, l_2=1$	1252	14	171	<b>10</b>	>1821	>32
9	$s=100, c=0.5, R=3, l_1=1, l_2=1$	31	<b>1</b>	16	2	136	9.6
10	$s=100, c=0.5, R=5, l_1=10, l_2=5$	42	<b>1.8</b>	22	4.1	178	16
11	$s=100, c=0.9, R=3, l_1=0, l_2=0$	>800	>27	99	<b>17</b>	>748	>60
12	$s=100, c=0.9, R=5, l_1=1, l_2=1$	1218	51	112	<b>26</b>	880	72

TABLE 3.3

Test Problem I. Number of iterations and time (in seconds) until accuracy measure  $|h(\mathcal{A}_R^{(i)}) - h^*|$  is reduced to  $10^{-10}$ . The smallest times appear in bold. For these high-accuracy tests, ALS is still fastest for all ‘easy’ cases ( $c = 0.5$ ), but N-GMRES is substantially faster for all the ‘difficult’ cases ( $c = 0.9$ ).

Tables 3.2 and 3.3 show convergence results when higher accuracy is required, with  $|h(\mathcal{A}_R^{(i)}) - h^*| < 10^{-6}$  and  $|h(\mathcal{A}_R^{(i)}) - h^*| < 10^{-10}$ , respectively. Table 3.2, for medium accuracy  $10^{-6}$ , shows that ALS is (not unexpectedly) still fastest for all ‘easy’ cases ( $c = 0.5$ ), but N-GMRES or N-CG are faster for most of the ‘difficult’ cases ( $c = 0.9$ ). Table 3.3, for high accuracy  $10^{-10}$ , confirms that ALS is still fastest for all ‘easy’ cases ( $c = 0.5$ ), but N-GMRES is substantially faster for all ‘difficult’ cases ( $c = 0.9$ ). Note also that N-GMRES is faster than N-CG for all of the ‘difficult’ cases, sometimes substantially.

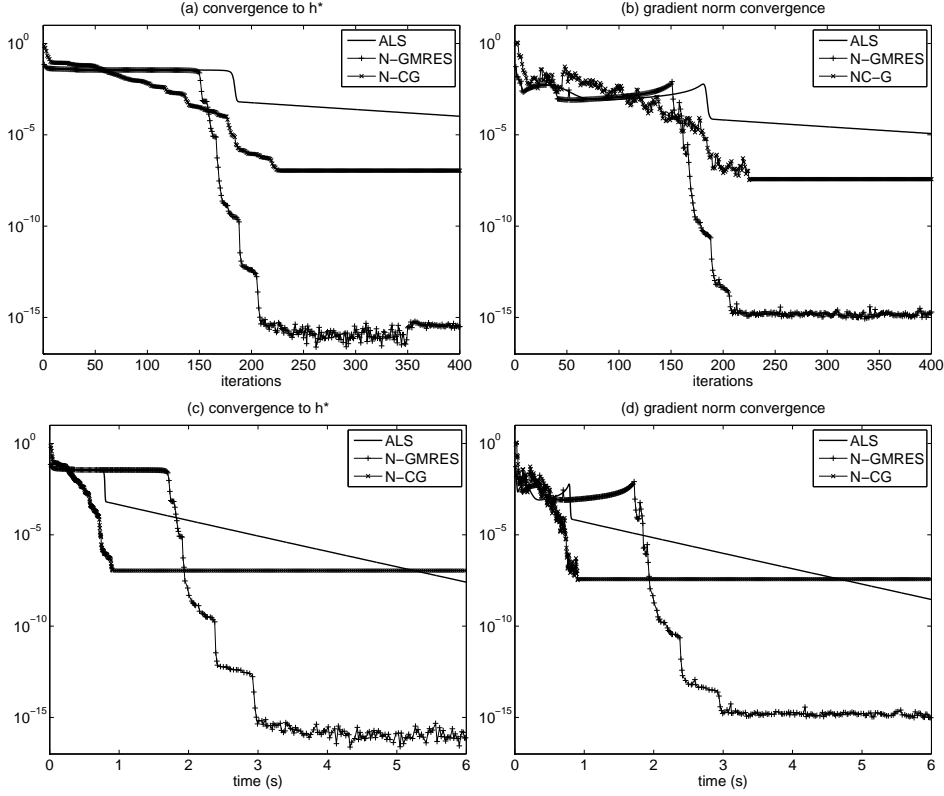


FIG. 3.2. *Test Problem I. Convergence plots for case 3 from Tables 3.1-3.3. Panel (b) shows that N-GMRES convergence kicks in only when ALS has gotten over a ‘bump’ in the gradient size. N-CG performs better in this case.*

It is instructive to consider convergence histories in some more detail for some of the test problems in Tables 3.1-3.3. Fig. 3.2 shows convergence plots for case 3 from Tables 3.1-3.3. For this problem (and the random initial guess used), convergence of N-GMRES only sets in after more than 150 iterations. Fig. 3.2(b) gives some indication as to why this is the case. N-GMRES accelerates ALS, but ALS needs to get over a ‘bump’ in the size of the gradient before it gets close enough to a stationary point for the N-GMRES acceleration to become effective. This points out a fundamental property of the N-GMRES acceleration procedure that is the topic of this paper: N-GMRES is not expected to offer a systematic improvement in the global convergence properties of the ‘preconditioning’ process it accelerates. The only potential help with global convergence would come from the line search in Step III of the algorithm, but this will only be effective when the search direction jointly determined by ALS and N-GMRES is suitable, and, as we argued before, we can only expect this to be the case consistently when the process approaches a stationary point. So N-GMRES still mainly relies on the preconditioning process to guide convergence on the global scale, and kicks in when close enough to a stationary point. In the case of Fig. 3.2, N-CG appears to perform better in getting close to a stationary point. Indeed, N-CG works in a way that is fundamentally different from N-GMRES: N-CG does not accelerate an underlying one-step method, but follows its own strategy for exploring the global

search space, with globalization provided by its line search. For the computation in Fig. 3.2, this leads to faster N-CG convergence.

However, we have observed that convergence behavior like in Fig. 3.2, while it may occur for some problem parameters and initial conditions, is not typical for Test Problem I with  $c = 0.9$ . Rather, convergence behavior as in Fig. 3.3, in which N-GMRES converges kicks in sooner, is more common (this is also indicated by the timing results in Tables 3.1-3.3). Fig. 3.3 shows convergence plots for case 11 from Tables 3.1-3.3. In this case, N-GMRES convergence kicks in early, and N-CG's strategy for global convergence does not appear successful. For high accuracy, N-GMRES is much faster in this case than both ALS and N-CG, and this is generally the case (for the 'difficult' problems with  $c = 0.9$ ), as confirmed by the numbers in Table 3.3. We recall that ALS convergence plots of an 'easy' case with  $c = 0.5$  are given in Fig. 1.1.

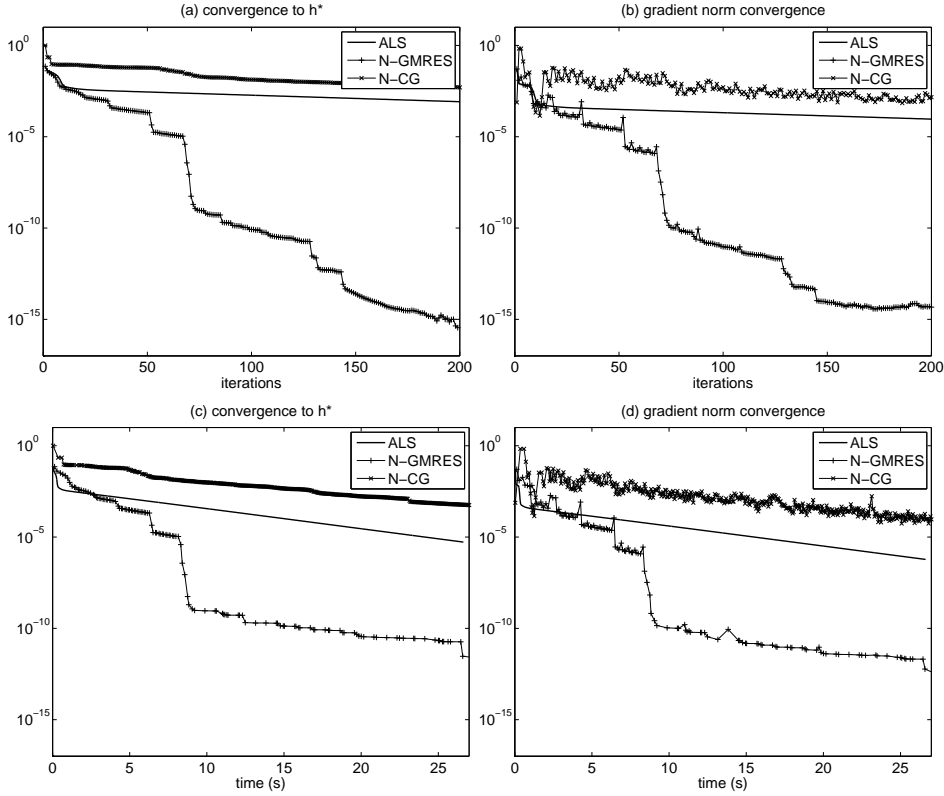


FIG. 3.3. *Test Problem I. Convergence plots for case 11 from Tables 3.1-3.3. N-GMRES convergence kicks in early, which leads to fast convergence, as is the case for most tests in Table 3.3 with  $c = 0.9$ .*

Fig. 3.2 shows another conspicuous difference in convergence behavior between N-GMRES and N-CG. N-CG convergence to a stationary point as measured by the size of the normalized gradient of the objective function (panel (b) in Fig. 3.2) stalls at a value of approximately  $10^{-7}$ . This is so because convergence of N-CG is limited by the accuracy of the line search, since it is the line search mechanism that ultimately determines the solution N-CG converges to. In principle this could be remedied by a more accurate line search, but this may incur extra cost, and we have found it difficult

to significantly increase the accuracy of the Moré-Thuente line search by changing its parameters. We have found that this gradient stalling occurs consistently for all problems we have considered; see also the other convergence plots in this paper. In the test of Fig. 3.2, this convergence stalling also limits the accuracy of the stationary point to about  $10^{-7}$ , but this is not always the case (in some of the forthcoming plots N-CG converges to machine accuracy in terms of the  $|h - h^*|$  measure). It is interesting to note that N-GMRES does not appear to suffer from this stalling in the gradient norm convergence. The likely explanation of this goes as follows: the N-GMRES procedure is such that, close to a stationary point, the accelerated iterates provided by N-GMRES may converge efficiently to the stationary point, even without the help of the globalizing line search. This is so because the linearization of (2.6) is expected to be highly accurate close to a stationary point (for problems that are sufficiently smooth), and the accelerated iterate is highly accurate by itself. N-GMRES does thus not need to rely on the line search mechanism for accuracy once near a stationary point, in contrast to N-CG, which relies on the line search for accurate convergence. This difference points to a potential advantage of N-GMRES over N-CG.

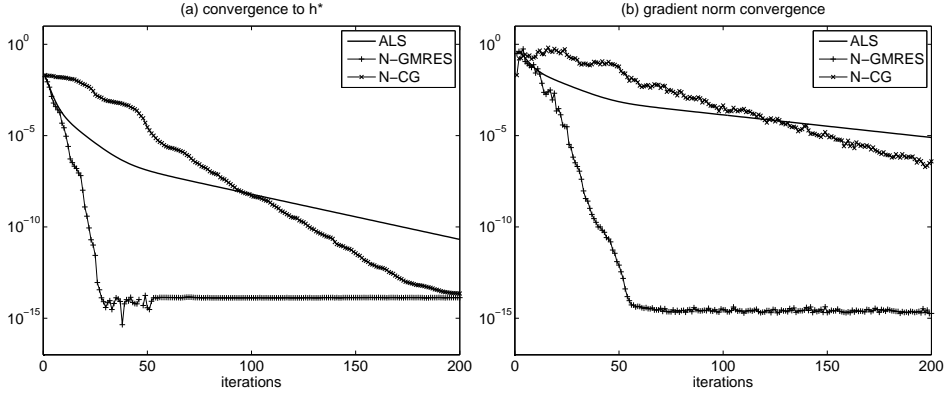


FIG. 4.1. *Test Problem II. Parameters are  $d = 3$  (and thus  $N = 6$ ),  $s = 6$  and  $R = 3$ , leading to a 6-way tensor of size  $6 \times 6 \times 6 \times 6 \times 6 \times 6$  for which an  $R = 3$  CP approximation is sought. Convergence plots for ALS, N-CG, and N-GMRES with window size  $w = 20$ .*

**4. Numerical Results: Sparse Tensor Test Problem.** In this section, we present numerical results for a simple sparse test problem:

**TEST PROBLEM II:** Standard finite difference Laplacian tensor on a regular grid of size  $s^d$  in  $d$  dimensions. This test problem results in an  $N$ -way sparse data tensor  $\mathcal{T}$  with  $N = 2d$  and nonzero elements of value  $2d$  and  $-1$ . For example, for  $d = 2$ ,  $\mathcal{T} \in \mathbb{R}^{s \times s \times s \times s}$ , and the nonzero tensor elements are  $t(i, j, i, j) = 2d = 4$ , for  $i = 1, \dots, s$  and  $j = 1, \dots, s$ , and  $t(i, j, i+1, j) = -1$ ,  $t(i, j, i, j+1) = -1$ ,  $t(i, j, i-1, j) = -1$ , and  $t(i, j, i, j-1) = -1$  (with the usual exceptions at boundary points of the  $d$ -dimensional regular grid).

This is an academic test problem, but it provides sparse tensors that are suitable for testing our numerical method.

As in the previous section, we first consider the effect of varying the N-GMRES window size. Fig. 4.1 shows convergence plots for an instance of Test Problem II with parameters  $d = 3$  and  $s = 6$ . Tensor  $\mathcal{T}$  is a 6-way tensor of size  $6 \times 6 \times 6 \times 6 \times 6 \times 6$ ,

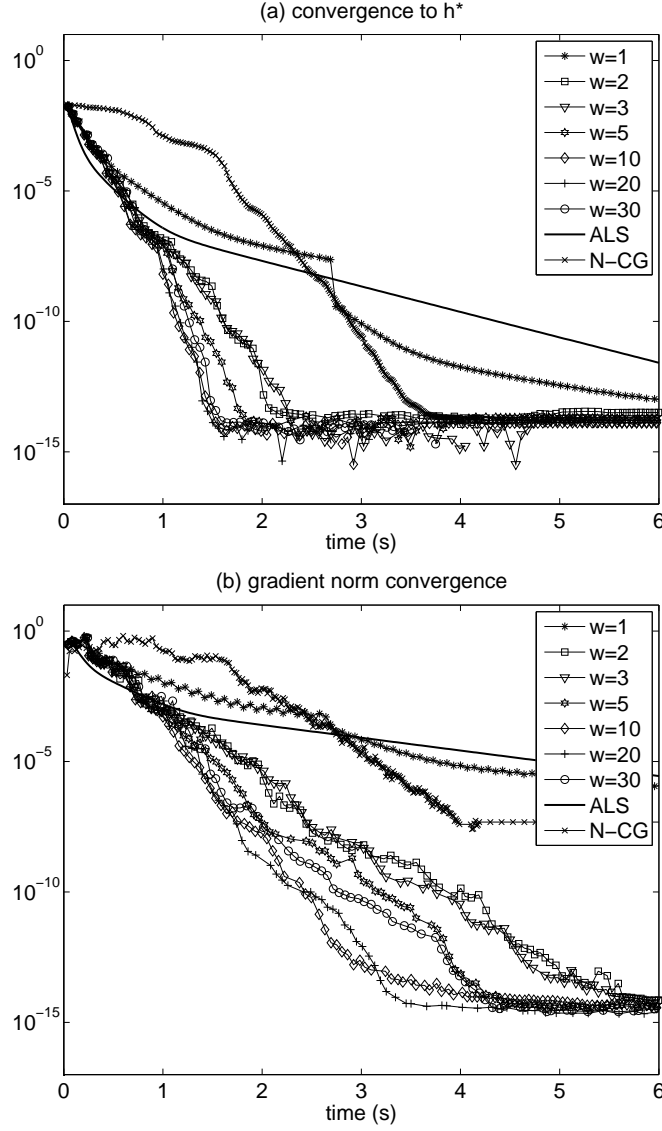


FIG. 4.2. Test Problem II ( $N = 6, s = 6, R = 3$ ). Convergence plots as a function of the  $N$ -GMRES window size,  $w$ . Window size  $w = 20$  emerges as a good choice for fast convergence when high accuracy is required.

and we seek a CP decomposition with  $R = 3$ . Fig. 4.1, with N-GMRES window size  $w = 20$ , shows that ALS is rather slow for this problem. N-GMRES significantly reduces the number of iterations, faster than N-CG.

Fig. 4.2 shows convergence plots as a function of time for this test problem, for varying window size  $w$ . For this test problem, as in the previous section, window size  $w = 20$  also appears a suitable choice, and we use it for the remaining numerical tests in this section.

Tables 4.1-4.3 show convergence results for a series of sparse Test Problem II runs with a wide range of parameter values  $N, s$  and  $R$ . We have found that the convergence

$h^*$ accuracy $10^{-3}$		ALS		N-GMRES		N-CG	
problem parameters		it	time	it	time	it	time
1	$N=4, s=8, R=6$	25	<b>0.66</b>	22	1.2	45	0.83
2	$N=4, s=8, R=6$	9	<b>0.26</b>	9	0.53	57	0.96
3	$N=4, s=16, R=3$	10	<b>0.15</b>	7	0.2	23	0.36
4	$N=4, s=16, R=3$	12	<b>0.19</b>	9	0.27	18	0.3
5	$N=6, s=4, R=2$	12	<b>0.22</b>	9	0.29	30	0.53
6	$N=6, s=4, R=2$	9	<b>0.17</b>	7	0.25	325	2.7
7	$N=6, s=8, R=5$	9	<b>0.37</b>	9	1.1	100	24
8	$N=6, s=8, R=5$	8	<b>0.37</b>	8	1.6	86	22
9	$N=8, s=4, R=2$	11	<b>0.34</b>	8	0.53	52	3.1
10	$N=8, s=4, R=2$	14	<b>0.42</b>	9	0.61	120	8.7

TABLE 4.1

Test Problem II. Number of iterations and time (in seconds) until accuracy measure  $|h(\mathcal{A}_R^{(i)}) - h^*|$  is reduced to  $10^{-3}$ . The smallest times appear in bold. ALS is fastest for these low-accuracy tests.

$h^*$ accuracy $10^{-6}$		ALS		N-GMRES		N-CG	
problem parameters		it	time	it	time	it	time
1	$N=4, s=8, R=6$	182	4.4	44	2.5	91	<b>1.4</b>
2	$N=4, s=8, R=6$	67	1.6	20	<b>1.2</b>	163	2.2
3	$N=4, s=16, R=3$	410	5.9	94	3	103	<b>1.3</b>
4	$N=4, s=16, R=3$	203	3	53	1.7	124	<b>1.4</b>
5	$N=6, s=4, R=2$	29	0.53	13	<b>0.44</b>	75	1
6	$N=6, s=4, R=2$	27	0.51	12	<b>0.45</b>	351	3
7	$N=6, s=8, R=5$	212	<b>8.3</b>	63	14	138	32
8	$N=6, s=8, R=5$	55	<b>2.2</b>	23	5.2	147	34
9	$N=8, s=4, R=2$	38	1.1	15	<b>1.1</b>	75	4.3
10	$N=8, s=4, R=2$	42	<b>1.3</b>	19	1.4	>280	>19

TABLE 4.2

Test Problem II. Number of iterations and time (in seconds) until accuracy measure  $|h(\mathcal{A}_R^{(i)}) - h^*|$  is reduced to  $10^{-6}$ . The smallest times appear in bold. N-GMRES and N-CG are competitive with ALS for these medium-accuracy tests.

behavior has more variation for Test Problem II than for Test Problem I. In particular, it happens more often that ALS, N-GMRES and N-CG converge to stationary points with different values of  $h$ , and the convergence profiles appear more sensitive to the initial guess. For this reason, we present two runs with different initial conditions for each parameter choice in Tables 4.1-4.3, which give a more representative idea of how the methods perform on average. In the tables, we only present cases for which the three methods converge to a stationary point with the same value of  $h$  (which still happens commonly).

Tables 4.1-4.3 largely confirm the observations we made for Test Problem I in the previous section. Table 4.1 shows that ALS is the fastest method for low accuracy computations. Table 4.2 shows that N-GMRES and N-CG become competitive for medium-accuracy computations, and Table 4.3 shows that N-GMRES is almost always faster than ALS for high accuracy, sometimes substantially. N-GMRES is generally also faster than N-CG.

It is again instructive to consider convergence histories in some more detail for

$h^*$ accuracy $10^{-10}$		ALS		N-GMRES		N-CG	
problem parameters		it	time	it	time	it	time
1	$N=4, s=8, R=6$	>400	>9.6	55	<b>3.1</b>	380	3.7
2	$N=4, s=8, R=6$	242	5.8	26	<b>1.5</b>	327	3.5
3	$N=4, s=16, R=3$	>800	>12	119	3.8	419	<b>3.5</b>
4	$N=4, s=16, R=3$	724	11	84	<b>2.7</b>	375	3.2
5	$N=6, s=4, R=2$	52	0.94	19	<b>0.65</b>	153	1.6
6	$N=6, s=4, R=2$	51	0.95	18	<b>0.67</b>	386	3.3
7	$N=6, s=8, R=5$	613	24	81	<b>18</b>	213	40
8	$N=6, s=8, R=5$	127	<b>5.1</b>	31	6.8	262	46
9	$N=8, s=4, R=2$	70	2	21	<b>1.5</b>	111	5.2
10	$N=8, s=4, R=2$	72	2.1	24	<b>1.8</b>	>280	>19

TABLE 4.3

Test Problem II. Number of iterations and time (in seconds) until accuracy measure  $|h(\mathcal{A}_R^{(i)}) - h^*|$  is reduced to  $10^{-10}$ . The smallest times appear in bold. N-GMRES outperforms ALS for most of these high-accuracy tests, and is normally also faster than N-CG.

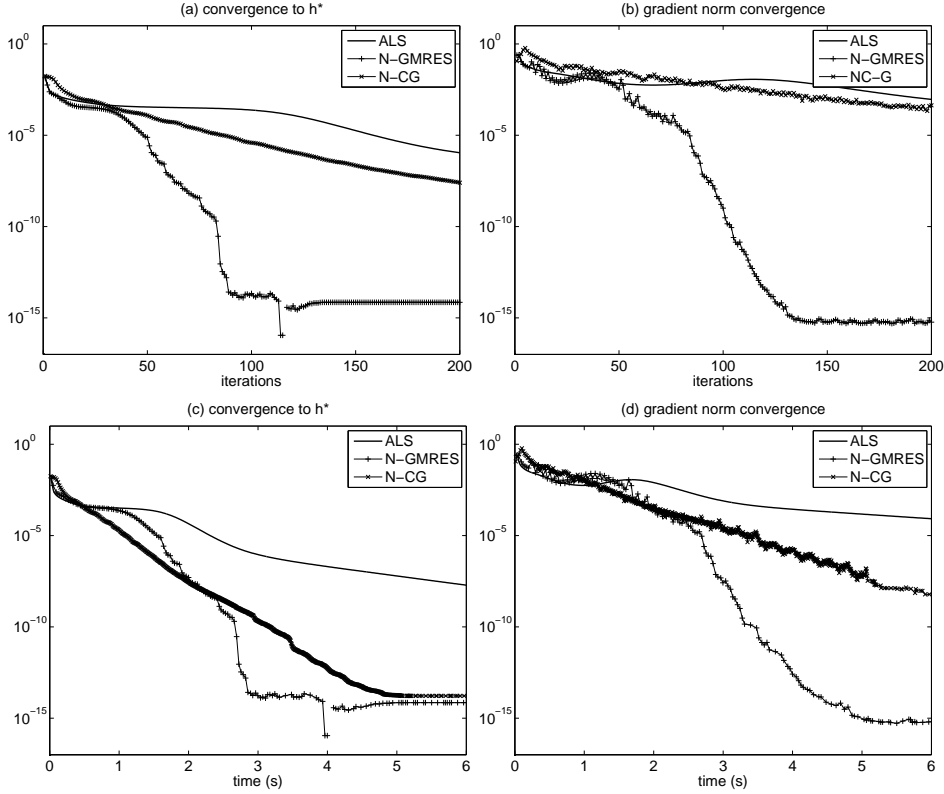


FIG. 4.3. Test Problem II. Convergence plots for case 4 from Tables 4.1-4.3.

some of the test problems in Tables 4.1-4.3. Fig. 4.3 shows convergence plots for case 4 from Tables 4.1-4.3. In this case, N-CG is fastest for medium accuracy, and N-GMRES for high accuracy. ALS converges rather slowly. Fig. 4.4 shows convergence plots for case 5 from Tables 4.1-4.3. In this case, ALS converges fast, but N-GMRES



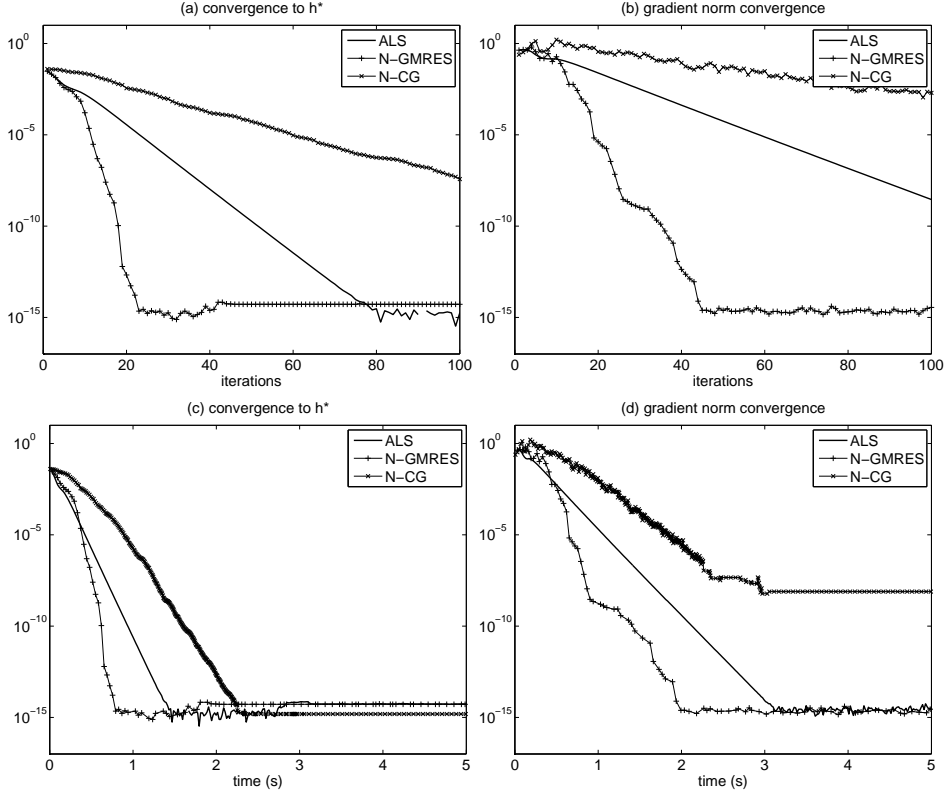


FIG. 4.4. *Test Problem II. Convergence plots for case 4 from Tables 4.1-4.3.*

acceleration is still able to improve its convergence speeds for medium and high accuracy. We see again that the gradient convergence of N-CG stalls around  $10^{-7}$ , but  $h$  does converge to  $h^*$  up to machine accuracy in this case.

**5. Conclusion.** We have presented a new optimization algorithm for computing a canonical rank- $R$  tensor approximation that has minimal distance to a given tensor in the Frobenius norm. The optimization algorithm uses a nonlinear version of GMRES iterate recombination that was proposed by Washio and Oosterlee for systems of nonlinear PDEs [18]. We apply this nonlinear GMRES acceleration to the ALS method, with the goal of efficiently driving the gradient of the CP objective function to zero, and combine it with a line search for globalization. The resulting 3-step N-GMRES optimization algorithm can be interpreted as an acceleration process of a one-step stand-alone method, or, alternatively, the stand-alone method can be considered as a preconditioning process for N-GMRES. We have explained how N-GMRES can be applied to the (approximate) canonical tensor decomposition problem.

Extensive numerical tests on dense and sparse tensors with varying sizes and dimensions (up to 8) show that N-GMRES with ALS preconditioning in many cases outperforms pure ALS when high accuracy is required, while ALS remains faster for low accuracy and ‘easy’ problems. N-GMRES is also competitive with the N-CG method studied in [1], and appears to outperform it significantly in some cases. In cases where ALS-preconditioned N-GMRES is slower than pure ALS, it is rarely so by much, and in the other cases the potential speed gain is very substantial. For this

reason, it may be a good strategy to put an N-GMRES acceleration wrapper around ALS if one does not know in advance whether ALS would converge slowly (which may depend on the problem and on the initial guess). Generally speaking, one may expect that adding an N-GMRES acceleration wrapper makes ALS convergence more robust.

It is a question of extensive current interest how Krylov methods can be generalized to tensor computations, see, for example, [16]. Our work presents the application of a nonlinear Krylov-type method to a tensor optimization problem, and we obtain acceleration that is significant in many cases. Our approach uses a nonlinear generalization of Krylov techniques, and it is perhaps not surprising that this seems to be a promising approach for tensor minimization problems, which are indeed inherently nonlinear (more specifically, multilinear).

A promising avenue for further research is to explore how the proposed N-GMRES optimization algorithm can be used to accelerate existing methods other than ALS for canonical tensor decomposition. Similarly, it would also be interesting to investigate how the N-GMRES optimization algorithm can accelerate ALS-type algorithms (or other algorithms) for other tensor approximation problems, for example, the best rank- $(R_1, R_2, R_3)$  approximation of a tensor, see, e.g., [8]. Furthermore, the nonlinear GMRES optimization algorithm proposed in this paper is based on general concepts and may be applied to other nonlinear optimization problems.

#### REFERENCES

- [1] E. ACAR, D.M. DUNLAVY, AND T.G. KOLDA, *A Scalable Optimization Approach for Fitting Canonical Tensor Decompositions*, Journal of Chemometrics, 25 (2011), pp. 67–86.
- [2] B.W. BADER AND T.G. KOLDA, *MATLAB Tensor Toolbox Version 2.4*, <http://csmr.ca.sandia.gov/~tgkolda/TensorToolbox/>, March 2010.
- [3] J.D. CARROLL AND J.J. CHANG, *Analysis of individual differences in multidimensional scaling via an N-way generalization of Eckart-Young decomposition*, Psychometrika, 35 (1970), pp. 283–319.
- [4] L. DE LATHAUWER, *A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization*, SIAM Journal on Matrix Analysis and Applications, 28 (2006), pp. 642–666.
- [5] L. DE LATHAUWER, B. DE MOOR, AND J. VANDEWALLE, *Computation of the canonical decomposition by means of a simultaneous generalized Schur decomposition*, SIAM Journal on Matrix Analysis and Applications, 26 (2004), pp. 295–327.
- [6] D.M. DUNLAVY, T.G. KOLDA, AND E. ACAR, *Poblano v1.0: A Matlab Toolbox for Gradient-Based Optimization*, Technical Report SAND2010-1422, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, March 2010.
- [7] R.A. HARSHMAN, *Foundations of the PARAFAC procedure: Models and conditions for an explanatory multi-modal factor analysis*, UCLA working papers in phonetics, 16 (1970), pp. 1–84.
- [8] M. ISHTEVA, L. DE LATHAUWER, P. ABSIL, AND S. VAN HUFFEL, *Differential-geometric Newton method for the best rank- $(R_1, R_2, R_3)$  approximation of tensors*, Numerical Algorithms, 51 (2009), pp. 179–194.
- [9] T.G. KOLDA AND B.W. BADER, *Tensor Decompositions and Applications*, SIAM Review, 51 (2009), pp. 455–500.
- [10] J.J. MORÉ AND D.J. THUENTE, *Line search algorithms with guaranteed sufficient decrease*, ACM Transactions on Mathematical Software, 20 (1994), pp. 286–307.
- [11] J. NOCEDAL AND S.J. WRIGHT, *Numerical optimization*, Springer, Berlin, 2006.
- [12] C.W. OOSTERLEE, *On multigrid for linear complementarity problems with application to American-style options*, Electronic Transactions on Numerical Analysis, 15 (2003), pp. 165–185.
- [13] C.W. OOSTERLEE AND T. WASHIO, *Krylov Subspace Acceleration of Nonlinear Multigrid with Application to Recirculating Flows*, SIAM J. Sci. Comput., 21 (2000), pp. 1670–1690.

- [14] M. RAJIB, P. COMON, AND RA HARSHMAN, *Enhanced line search: A novel method to accelerate PARAFAC*, SIAM Journal on Matrix Analysis and Applications, 30 (2008), pp. 1128–1147.
- [15] Y. SAAD AND M.H. SCHULTZ, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Comp., 7 (1986), pp. 856–869.
- [16] B. SAVAS AND L. ELDÉN, *Krylov-Type Methods for Tensor Computations*, submitted (2010), arXiv:1005.0683v2.
- [17] G. TOMASI AND R. BRO R, *A comparison of algorithms for fitting the PARAFAC model*, Computational Statistics and Data Analysis, 50 (2006), pp. 1700–1734.
- [18] T. WASHIO AND C.W. OOSTERLEE, *Krylov subspace acceleration for nonlinear multigrid schemes*, Electronic Transactions on Numerical Analysis, 6 (1997), pp. 271–290.